REMARKS

Claims 1-37 stand rejected. Claims 38-40 have been added. Applicant is canceling claims 14, 21- 23, and 29-31 without prejudice or disclaimer. Claims 1, 20 and 28 have been amended to clarify the features of the invention. As a result, claims 1-13, 15-20, 24-28, and 32-40 are pending for examination with claims 1, 20 and 28 being independent claims. The amendments made find support in the specification and do not constitute new matter.

Claims 1-37 stand rejected under 35 U.S.C. §103(a) as being unpatentable overBaisley et al. U.S. Patent 6,330,569 B1 ("Baisley") in view of Ronning et al. U.S. 2003/0195974 A1 ("Ronning"). The Examiner states that "it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Baisely and Ronning because, being able to update executables would enable more dynamic updating of the program."

Applicants have amended Claim 1 to call for:

> "a whole name matching at least one existing executable program;
>
> a program matching criteria matching at least one existing executable program; and
>
> a reference to at least one substitute program segment in the set of substitute program segments; and
>
> an index including a set of identifiers, wherein each identifier identifies one of the set of program entries, and wherein each identifier comprises a name portion of the whole name"
>
> (underlining added for emphasis)

Applicants have amended Claim 20 to call for:

"multi-tiered matching of identification information for the particular executable program to a program matching criteria for an entry within the set of program entries, the multi-tiered matching step comprising the sub-steps of:

first executing a first search on a name portion of a whole name of an index having identifying information for each one of the set of program entries to identify a first set of potential matching entries;

second executing a second search on at least a portion of the first set of potential matching entries to identify a second set of potential matching entries matching the particular executable program based upon a whole name for the program entry; and

third executing a third search on at least a portion of the second set of potential matching entries to identify a program entry matching the particular program based upon the program matching criteria for the program entry"

(underlining added for emphasis)

Applicants have amended Claim 28 to call for:

"multi-tiered matching identification information for the particular executable program to a program matching criteria for an entry within the set of program entries, the multi-tiered matching step comprising the sub-steps of:

first executing a first search on a name portion of an index having identifying information for each one of the set of program entries to identify a first set of potential matching entries;

second executing a second search on at least a portion of the first set of potential matching entries to identify a second set of potential matching entries matching the particular executable program based upon a whole name for the program entry; and

third executing a third search on at least a portion of the second set of potential matching entries to identify a program entry matching the particular program based upon the program matching criteria for the program entry"

(underlining added for emphasis)

Applicants submit that the invention as claimed in Claims 1, 20 and 28 are neither taught, described nor suggested in Baisley even in view of Ronning.

The present invention provides: "Next, during step 406 the Get Match API 206 searches the index 204 for the name, or portion of the name, of the executable program provided during step 400. By way of example, the Get Match API 206 compares the first eight characters of the provided executable program name to each entry within the index 204. The index 204 typically only occupies around 10 kilobytes of memory space, and is located at the beginning of the database, and therefore it can be read in one read operation. At step 408, if no matches are found in the index 204, then control passes to step 410 wherein the Get Match API 206 returns a message to the calling process that no matches were found (i.e., this program does not have a corresponding entry in the modification specifications 200). Control then passes to the End.

If however during step 408 at least one matching index entry is located by the Get Match API 206 within the index 204, then control passes to step 412. During step 412, the Get Match API 206 completes a second tier search for potential matching executable (i.e., <EXE> tagged) entries within the modification specifications 200. During step 412 the Get Match API 206 compares the name of the executable program (provided during step 400) to whole names of programs stored within particular individual EXE tagged entries of the modification specifications 200.

...

If however during step 414 one or more whole names match the file name, then control passes to step 418. During step 418, the Get Match API 206 performs a matching operation for a first potential matching entry rendered during step 412. A criteria listed for a first potential matching entry in the database is checked against the executable program and files located in a

specified directory and/or subdirectory. The matching operation is dictated by the matching criteria specified by the potential matching entry. An entry's criteria can include, but is not limited to: a file name, a file size, a file checksum, file version information, and file creation time." (underlining added for emphasis)

Baisley, on the other hand, provides: "Referring now to FIG. 5A , the first of a three-sheet diagram of a flow chart for the overall process of the present invention. The process begins with a start bubble 51 followed by a step beginning a depth-first traversal of the object tree (block 52). Next, an inquiry is made as to whether or not the traversal is complete (diamond 53). If the answer to this inquiry is yes, then the process ends. On the other hand, if the answer is no, then the next object is retrieved from the object tree and it becomes the 'Current-Object' (block 54). For each 'Current-Object', Attributes are matched to corresponding Attributes for the repository object (block 55). Next, an inquiry is made as to whether or not all Attributes match (diamond 56). If the answer to this inquiry is no, then the repository object is reserved (block 57) and changes are made to the ghost object Attributes to match Attributes found in XML objects (block 58). The step of reserving repository objects, i.e., block 57, will be amplified in FIG. 6 and further explained hereinbelow.

...

Referring now to FIG. 5C at the connector B, all objects owned by the XML objects at the current depth are retrieved from the XML file (block 67). Next, an inquiry is made as to whether or not the repository object has the same number of owned objects (diamond 68). If the answer to this inquiry is yes, then another inquiry is made as to whether or not all the owned objects match (diamond 69). If the answer to the inquiry in either the diamond 68 or the diamond 69 is no, then the repository object is reserved (block 70). After this, changes are made to the repository ghost object owned elements to match the owned elements of the XML objects (block 771. Upon completion of this step, or if the answer to the inquiry in the diamond 69 is yes, a return is made back to the diamond 53 to determine whether or not the traversal of the object tree is complete." (see figs 5A - 5C and col. 6, line 23 thru col. 7, line 13; underlining added for emphasis)

Accordingly, the Applicants submit that Claims 1, 20 and 28 are not unpatentable over Baisley in view of Ronning.

Claims 2-13 and 15-19 are dependent on Claim 1; as such, these dependent claims are believed allowable based upon Claim 1.

Claims 24-27 are dependent on Claim 20; as such, these dependent claims are believed allowable based upon Claim 20.

Claims 32-37 are dependent on Claim 28; as such, these dependent claims are believed allowable based upon Claim 28.

Claims 38-40 have been added to further define the invention.

CONCLUSION

Accordingly, in view of the above amendment and remarks it is submitted that the claims are patentably distinct over the prior art and that all the rejections to the claims have been overcome. Reconsideration and reexamination of the above Application is requested. Based on the foregoing, Applicant respectfully requests that pending claims be allowed, and that a timely Notice of Allowance be issued in this case. If the Examiner believes, after this amendment, that the application is not in condition for allowance, the Examiner is requested to call the Applicant's attorney at the telephone number listed below.

If this response is not considered timely filed and if a request for an extension of time is otherwise absent, Applicant hereby requests any necessary extension of time. If there are any fees due in connection with the filing of this amendment, please charge the fees to our Deposit Account No. 50-0463.

In the event that there are any outstanding matters remaining in the filing of this amendment, the Examiner is invited to contact the undersigned to discuss this application.

Respectfully submitted,

Date: 2/15/05

By: PAUL HEYNSSENS

Microsoft Corporation
One Microsoft Way
Redmond WA 98052-6399

Paul B. Heynssens
Registration Number: 47,648
Direct telephone: (425) 707-3913

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]

I hereby certify that this correspondence is being:

☒ deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop __RCE__, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450

☐ transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.

2/15/05
Date

Signature

PAUL HEYNSSENS
Type or Print Name